**Assigning RNA-DNA interactions**

#${file} is the file name

bedtools bamtobed -i ${file}\_filter.gDNA.bam > ${file}\_DNA.bed

#obtain reads’ binding loci and constribution score

samtools view -F 4 -S ${file}\_filter.gDNA.bam |awk 'BEGIN{FS=OFS="\t"}{if($17!="XZ:f:0"){print $1,$2,$3,$4,$17}}' | sed 's/XZ\:f\://g' | awk 'BEGIN{FS=OFS="\t"}{if($2==0 || $2==256){print $0,"+"} else if($2==16 || $2==272){print $0,"-"}}' >${file}\_readNameAndLoci

#Annotated the constribution score to the Bed file

awk 'BEGIN{FS=OFS="\t"}NR==FNR{A[$1"%%"$3"%%"($4-1)"%%"$6]=$5} NR>FNR{$7=A[$4"%%"$1"%%"$2"%%"$6];if($7!=""){print $1,$2,$3,$4,$7,$6}}' ${file}\_readNameAndLoci ${file}\_DNA.bed > ${file}\_DNAaddScore.bed

#Obtain read name and all the mapping loci, considering the multiple mapped reads

awk 'BEGIN { OFS = "\t" } {if(K[$4]) K[$4]=K[$4]";"$1"%%"$2"%%"$3"%%"$6"%%"$5; else K[$4]=$1"%%"$2"%%"$3"%%"$6"%%"$5;}END{for(x in K) print x, K[x]}' ${file}\_DNAaddScore.bed > ${file}\_DNA\_use

#Annotated DNA read to AluI DNA bin

intersectBed -wo -f 1.0 -a ${file}\_DNAaddScore.bed -b /home/haoyj/tasks/Heshunmin/haoyj\_fruitfly\_chrAluI/dm6/chrCombine.bed > ${file}.DNAIntersectDNABin

#Remove PCR duplicates

cut -f 1 ${file}\_DNA\_use |sed 's/\_/\t/g'|cut -f 3|sort|uniq -c|sed 's/^[ \t]\*//g'|sed 's/ /\t/g' > ${file}\_number\_primer

#The same pipeline to processing the RNA part

bedtools bamtobed -i ${file}\_filter.cDNA.bam | awk 'BEGIN{OFS="\t"} {$6 = $6=="+"? "-":"+"; print}' > ${file}\_cDNA.bed

samtools view -F 4 -S ${file}\_filter.cDNA.bam|awk 'BEGIN{FS=OFS="\t"}{if($17!="XZ:f:0"){print $1,$2,$3,$4,$17}}' |sed 's/XZ\:f\://g' |awk 'BEGIN{FS=OFS="\t"}{if($2==0 || $2==256){print $0,"-"}else if($2==16 || $2==272){print $0,"+"}}' >${file}\_readNameAndLoci

awk 'BEGIN{FS=OFS="\t"}NR==FNR{A[$1"%%"$3"%%"($4-1)"%%"$6]=$5}NR>FNR {$7=A[$4"%%"$1"%%"$2"%%"$6]; if($7!=""){print $1,$2,$3,$4,$7,$6}}' ${file}\_readNameAndLoci ${file}\_cDNA.bed > ${file}\_cDNAaddScore.bed

awk 'BEGIN { OFS = "\t" } {if(K[$4]) K[$4]=K[$4]";"$1"%%"$2"%%"$3"%%"$6"%%"$5; else K[$4]=$1"%%"$2"%%"$3"%%"$6"%%"$5;}END{for(x in K) print x, K[x]}' ${file}\_cDNAaddScore.bed > ${file}\_RNA\_use

#anno RNA to transcript

intersectBed -wo -f 1.0 -split -s -a ${file}\_cDNAaddScore.bed -b Drosophila\_melanogaster.BDGP6.88.chr.bed > ${file}.cDNAIntersectTranscript

cut -f 1 ${file}\_RNA\_use |sed 's/\_/\t/g'|cut -f 3|sort|uniq -c|sed 's/^[ \t]\*//g'|sed 's/ /\t/g' > ${file}\_number\_primer

#Obtain the RNA and DNA pairs according to the read name

awk 'BEGIN{FS=OFS="\t"}NR==FNR{A[$1]=$2} NR>FNR{$3=A[$1]; if($3!="") {print} } ' ../${file}\_gDNA/${file}\_DNA\_use ../${file}\_cDNA/${file}\_RNA\_use >${file}.RNA\_DNA

#Remove PCR amplification results

awk 'BEGIN{FS=OFS="\t"}{if(K[$2"@@"$3]){K[$2"@@"$3]=K[$2"@@"$3]";"$1}else{K[$2"@@"$3]=$1}}E ND{for (i in K){print i,K[i]}}' ${file}.RNA\_DNA > ${file}.RNA\_DNA\_lociAndName

awk'BEGIN{FS=OFS="\t"}{if(A[$2"&&"$3]){A[$2"&&"$3]=A[$2"&&"$3]";"$1}else{A[$2"&&"$3]=$1}}END{for(i in A){print i,A[i]}}' ${file}.RNA\_DNA >${file}.RNA\_DNA\_lociCombine

python rmPCRduplicateandNNN.py ${file}.RNA\_DNA\_lociCombine ${file}.RNA\_DNA\_locirmPCR

awk 'BEGIN{FS=OFS="\t"}{split($3,a,";");for (i in a){if(a[i]!=""){print a[i],$1}}}' ${file}.RNA\_DNA\_locirmPCR |sed 's/&&/\t/g' > ${file}.RNA\_DNA\_locirmPCR\_use

awk 'BEGIN{FS=OFS="\t"}{split($2,a,";");split($3,b,";");for (i in a){ for (j in b) print $1,a[i],b[j]}}' ${file}.RNA\_DNA\_locirmPCR\_use |sed 's/%%/\t/g' >${file}.RNA\_DNA\_locirmPCR\_use\_pair

#Add transcript and AluI DNAbin information to the RNA-DNA pair

awk 'BEGIN{FS=OFS="\t"}NR==FNR{A[$1"%%"$2"%%"$3"%%"$4"%%"$5"%%"$6]=$10}NR>FNR {$12=A[$2"%%"$3"%%"$4"%%"$1"%%"$6"%%"$5];if($12!=""){print}}' ${file}.cDNAIntersectTranscript ${file}.RNA\_DNA\_locirmPCR\_use\_pair > ${file}.RNA\_DNA\_locirmPCR\_use\_pairaddT

awk 'BEGIN{FS=OFS="\t"}NR==FNR{A[$1"%%"$2"%%"$3"%%"$4"%%"$5"%%"$6]=$10}NR>FNR {$13=A[$7"%%"$8"%%"$9"%%"$1"%%"$11"%%"$10];if($13!=""){print}}' ${file}.DNAIntersectDNABin ${file}.RNA\_DNA\_locirmPCR\_use\_pairaddT > ${file}.RNA\_DNA\_locirmPCR\_use\_pairaddTaddB

#Add interactions' supported read counts

awk'BEGIN{FS=OFS="\t"}{if(K[$12"%%"$13]){K[$12"%%"$13]=K[$12"%%"$13]+$6\*$11}else {K[$12"%%"$13]=$6\*$11}}END{for (i in K){print i,K[i]}}' ${file}.RNA\_DNA\_locirmPCR\_use\_pairaddTaddB > ${file}.RNA\_DNA\_locirmPCR\_addScore

# Annotated the RNA part to repeat sequences similar to the process which annotated the RNA part to the transcript sequences described above.

**Construction of non-specific background using mixed GRID-seq libraries**

#Annotated the mix reads to the 1kb bins

intersectBed -wo -f 1.0 -a ../mix2\_DNAaddID.bed -b dm6\_normalchr\_1k\_use.bed > mix2.DNAIntersect1kBin

awk 'BEGIN { FS=OFS = "\t" } {if(K[$10]) K[$10]=K[$10]+$5; else K[$10]=$5;}END{for(x in K) print x, K[x]}' mix2.DNAIntersect1kBin >mix2.1kBinandScore

# Normalized the mix reads to per million

awk 'BEGIN{FS=OFS="\t"}NR==FNR{A[$1]=($2\*1000000)/(num)}NR>FNR {$5=A[$4];if($5=="") {$5=0};print}' mix2.1kBinandScore dm6\_normalchr\_1k\_use.bed > dm6\_normalchr\_1k\_addNormScore.bed

# Smoothed by a moving window that includes 5 upstream and 5 downstream bins

for i in chr2L chr2R chr3L chr3R chr4 chrX chrY;do

#the length is the bin number

if [ $i == "chr2R" ]

then

length=25287

elif [ $i == "chr2L" ]

then

length=23514

elif [ $i == "chr3R" ]

then

length=32080

elif [ $i == "chr3L" ]

then

length=28111

elif [ $i == "chr4" ]

then

length=1349

elif [ $i == "chrX" ]

then

length=23543

elif [ $i == "chrY" ]

then

length=3668

fi

grep "${i}" dm6\_normalchr\_1k\_addNormScore.bed > dm6\_normalchr\_1k\_addNormScore\_${i}.bed

python addZero\_10.py dm6\_normalchr\_1k\_addNormScore\_${i}.bed ${length} > dm6\_normalchr\_1k\_addNormScore\_${i}\_smooth.bed

paste dm6\_normalchr\_1k\_addNormScore\_${i}.bed dm6\_normalchr\_1k\_addNormScore\_${i}\_smooth.bed|cut -f 1-4,7 > smooth\_${i}.bed

awk 'BEGIN{FS=OFS="\t"}{for(i=$2;i<$3;i++){print $1,i,i+1,$5/1000}}' smooth\_${i}.bed > smooth\_${i}.use

intersectBed -wo -f 1.0 -a smooth\_${i}.use -b chrCombine.bed > smooth\_${i}\_InterDNABin.use

awk 'BEGIN{FS=OFS="\t"}{if(A[$8]){A[$8]=A[$8]+$4}else{A[$8]=$4}}END{for (i in A){print i,A[i]}}' smooth\_${i}\_InterDNABin.use >smooth\_${i}\_InterDNABinScore

done

cat smooth\_chr2L\_InterDNABinScore smooth\_chr2R\_InterDNABinScore smooth\_chr3L\_InterDNABinScore smooth\_chr3R\_InterDNABinScore smooth\_chr4\_InterDNABinScore smooth\_chrX\_InterDNABinScore smooth\_chrY\_InterDNABinScore > smooth\_combine\_InterDNABinScore

awk 'BEGIN{FS=OFS="\t"}NR==FNR{A[$1]=$2}NR>FNR{$3=A[$1];print $0,($2\*1000)/$3}' chrBinandLen smooth\_combine\_InterDNABinScore > smooth\_combine\_InterDNABinScore\_nomalized

**Filtering singular and background to identify specific RNA-DNA interactions**

# Filter non-specific RNA-DNA interactions

# Step1 is to remove the interactions supported different reads number below 2

cut -f 1,12,13 ${file}.RNA\_DNA\_locirmPCR\_use\_pairaddRepaddB| awk 'BEGIN { FS=OFS = "\t" } {if(K[$2"%%"$3]){ K[$2"%%"$3]= K[$2"%%"$3]+1}else{K[$2"%%"$3]=1}}END{for(x in K) {if(K[x]>=2){print x}}}' |sed 's/%%/\t/g'|cut -f 1,3|sort|uniq > ${file}.RNA\_DNA\_rm2RN

# Step 2 is to filter interactions which supported reads number significantly higher than whole genome background

sed 's/%%/\t/g' ${file}.RNA\_DNA\_locirmPCR\_addScore |awk 'BEGIN{FS=OFS="\t"}{if(K[$1]){K[$1]=K[$1]+$4}else{K[$1]=$4}}END{for (i in K){print i,K[i]}}' > ${file}.RepeatAndScore

awk 'BEGIN{FS=OFS="\t"}{print $0,($2\*1000)/137567477}' ${file}.RepeatAndScore > ${file}.RepeatAndScoreAddLMD

awk 'BEGIN{FS=OFS="\t"}NR==FNR{A[$1]=$2}NR>FNR{$3=A[$1];print}' RepeatNameandLength\_new ${file}.RNA\_DNA\_rm2RN >${file}.RNA\_DNA\_rm2RNaddRlen

awk 'BEGIN{FS=OFS="\t"}NR==FNR{A[$1]=$2}NR>FNR{$4=A[$2];print}' chrBinandLen ${file}.RNA\_DNA\_rm2RNaddRlen >${file}.RNA\_DNA\_rm2RNaddRlenaddBlen

sed 's/%%/\t/g' ${file}.RNA\_DNA\_locirmPCR\_addScore|awk 'BEGIN{FS=OFS="\t"} {if(A[$1"%%"$3]) {A[$1"%%"$3]=A[$1"%%"$3]+$4}else{A[$1"%%"$3]=$4}}END{for(i in A){print i,A[i]}}' |sed 's/%%/\t/g' > ${file}.RNA\_DNA\_locirmPCR\_addScore\_use

awk 'BEGIN{FS=OFS="\t"}NR==FNR{A[$1"%%"$2]=$3}NR>FNR{$5=A[$1"%%"$2];print}' ${file}.RNA\_DNA\_locirmPCR\_addScore\_use ${file}.RNA\_DNA\_rm2RNaddRlenaddBlen > ${file}.RNA\_DNA\_rm2RNaddRlenaddBlenaddScore

awk 'BEGIN{FS=OFS="\t"}NR==FNR{A[$1]=$3}NR>FNR{$6=A[$1];$7=($5\*1000)/$4;print}' ${file}.RepeatAndScoreAddLMD ${file}.RNA\_DNA\_rm2RNaddRlenaddBlenaddScore > ${file}.RNA\_DNA\_rm2RNAaddRlenaddBlenaddScoreaddLMD

python calculatePvalue.py ${file}.RNA\_DNA\_rm2RNAaddRlenaddBlenaddScoreaddLMD > ${file}.RNA\_DNA\_rm2RNAaddRlenaddBlenaddScoreaddLMDaddPvalue

awk 'BEGIN{FS=OFS="\t"}{if($7/$6>=2 && $8<=0.05){print}}' ${file}.RNA\_DNA\_rm2RNAaddRlenaddBlenaddScoreaddLMDaddPvalue > ${file}.RNA\_DNA\_rm2RNAaddRlenaddBlenaddScoreaddLMDaddPvalue\_filter

# Step3 is to remove the interactions which supported reads number significantly higher than the non-specific background

# smooth\_combine\_InterDNABinScore\_nomalized is the normalized the background according to the mix library

awk 'BEGIN{FS=OFS="\t"}NR==FNR{A[$1]=$4}NR>FNR{$9=A[$2];if($9==""){$9=0}; $10=($7\*1000000000)/($6\*137567477); print}' smooth\_combine\_InterDNABinScore\_nomalized ${file}.RNA\_DNA\_rm2RNAaddRlenaddBlenaddScoreaddLMDaddPvalue\_filter > ${file}.addNonspecificLMD

python calculatePvalue.py ${file}.addNonspecificLMD > ${file}.addNonspecificLMD\_pvalue

awk 'BEGIN{FS=OFS="\t"}{if( ($9!=0 && $10/$9>=2 && $11<=0.05) || ($9==0)){print}}' ${file}.addNonspecificLMD\_pvalue > ${file}.addNonspecificLMD\_pvalue\_filter

#Three python scripts used before.

#rmPCRduplicateandNNN.py

!/usr/bin/python

#0

Copyright (c) 2008 Yajing Hao <yahao@ucsd.edu>

This code is free software; you can redistribute it and/or modify it

under the terms of the BSD License (see the file COPYING included with

the distribution).

@status: experimental

@version: $Revision$

@author: Yajing Hao

@contact: yahao@ucsd.edu

"""

# ------------------------------------

# python modules

# ------------------------------------

import sys

import string

import os

#from wbed import ColumnReader

from Bio.Seq import Seq

from Bio.Alphabet import IUPAC

# ------------------------------------

# Main

# ------------------------------------

if \_\_name\_\_=="\_\_main\_\_":

if len(sys.argv)==1:

print "Usage: "+sys.argv[0]+" file1 is the input file,file2 is the output file name."

else:

fh=open(sys.argv[1])

newfile=open(sys.argv[2],"w")

for line in fh:

line=line.rstrip(os.linesep)

line=line.split("\t")

primerLoci=[]

readLeft=[]

readlist=line[1].split(";")

if (len(readlist)==1):

newfile.write(line[0]+"\t"+line[1]+"\t"+line[1]+";\n")

else:

for i in range(len(readlist)):

readFrag=readlist[i].split("\_")

if (readFrag[2] not in primerLoci) and readFrag[2]!="NNN":

readLeft.append(readlist[i])

primerLoci.append(readFrag[2])

newfile.write(line[0]+"\t"+line[1]+"\t")

for j in range(len(readLeft)):

newfile.write(readLeft[j]+";")

newfile.write("\n")

#calculatePvalue.py

!/usr/bin/python

#0

Copyright (c) 2008 Yajing Hao <yahao@ucsd.edu>

This code is free software; you can redistribute it and/or modify it

under the terms of the BSD License (see the file COPYING included with

the distribution).

@status: experimental

@version: $Revision$

@author: Yajing Hao

@contact: yahao@ucsd.edu

"""

# ------------------------------------

# python modules

# ------------------------------------

import sys

import string

import os

#from wbed import ColumnReader

from Bio.Seq import Seq

from Bio.Alphabet import IUPAC

from scipy.stats import poisson

import matplotlib.pyplot as plt

# ------------------------------------

# Main

# ------------------------------------

if \_\_name\_\_=="\_\_main\_\_":

if len(sys.argv)==1:

print "Usage: "+sys.argv[0]+" file1 file2..."

else:

fh=open(sys.argv[1])

for line in fh:

line=line.rstrip(os.linesep)

line1=line.split("\t")

#print line1[7]

pvalue=1-poisson.cdf(float(line1[6]), float(line1[5]))

print line+"\t"+str(pvalue)

#addZero\_10.py

!/usr/bin/python

#0

Copyright (c) 2008 Yajing Hao <yahao@ucsd.edu>

This code is free software; you can redistribute it and/or modify it

under the terms of the BSD License (see the file COPYING included with

the distribution).

@status: experimental

@version: $Revision$

@author: Yajing Hao

@contact: yahao@ucsd.edu

"""

# ------------------------------------

# python modules

# ------------------------------------

import sys

import string

import os

#from wbed import ColumnReader

from Bio.Seq import Seq

from Bio.Alphabet import IUPAC

from scipy.stats import poisson

import matplotlib.pyplot as plt

import numpy as np

import time

# ------------------------------------

# Main

# ------------------------------------

if \_\_name\_\_=="\_\_main\_\_":

if len(sys.argv)==1:

print "Usage: "+sys.argv[0]+" inputfile chrLength..."

else:

fh=open(sys.argv[1])

result=[]

#chr is the length of the specific chromatin

chr=int(sys.argv[2])

dicname={}

#print time.time()

num=1

for line in fh.readlines():

name=(line.split("\t"))[3]

temp=(line.split("\t"))[4].split("\n")[0]

result.append(list(map(float,temp.split(','))))

dicname[num]=name

num=num+1

#print time.time()

for i in range(1,chr+1):

if (i-5<=0):

mean=np.mean(result[0:i+5])

elif(i+5>(chr)):

mean=np.mean(result[i-6:chr])

else:

mean=np.mean(result[i-6:i+5])

print dicname[i]+"\t"+str(mean)